Planning and scheduling in the digital factory

Tamás Kis and Márton Drótos

Abstract Production planning and scheduling with the aid of software tools in today's manufacturing industries have become a common practice which is indispensable for providing high level customer service, and at the same time to utilize the production resources, like workforce, machine tools, raw materials, energy, etc., efficiently. To meet the new requirements, problem modeling tools, optimization techniques, and visualization of data and results have become part of the software packages. In this chapter some recent developments in problem modeling and optimization techniques applied to important and challenging industrial planning and scheduling problems are presented. We will focus on new problem areas which are still at the edge of current theoretical research, but they are motivated by practical needs. On the one hand, we will discuss project based production planning, and on the other hand, we will tackle a resource leveling problems in a machine environment. We will present the problems, some modeling and solution approaches, and various extensions and applications.

1 Introduction

In this chapter we overview some of the recent developments in automatic planning and scheduling of complex manufacturing processes. We will consider problems that in our experience frequently occur in practice, but they are not so well studied like several single machine scheduling problems, or the makespan minimization problem in job shops [5], [6].

Tamás Kis
Institute for Computer Science and Control, Kende str. 13-17, Budapest, Hungary H-1111, e-mail: kis.tamas@sztaki.mta.hu

Márton Drótos
Institute for Computer Science and Control, Kende str. 13-17, Budapest, Hungary H-1111, e-mail: drotos.marton@sztaki.mta.hu

We will introduce two problem areas in detail, and for each problem area we provide a problem formulation, some theoretical background, sketch a solution approach and summarize computational results. We will also provide references to the literature offering further results and extensions.

2 Project based production planning

Traditionally, production planning is concerned with determining production quantities of final products and that of their subcomponents over time based on manufacturing lead times and the bill-of-materials. This approach may be inadequate when dealing with a large variety of products manufactured in small quantities. For instance, when a customer orders e.g., a complex product made of several components, which have to be designed, manufactured, assembled, tested, and finally delivered to the customer, then determining production quantities is just not the right way of making a feasible plan, not mentioning that a manufacturing company may have several big projects running concurrently, which have to be controlled separately.

In order to build a planning model, we will use the terminology of project scheduling [8]. A project consists of *activities* needing various *resources*, and connected by *precedence constraints*. Consider for instance a manufacturing firm which produces complex products for its customers. Each customer order becomes a project, where an activity represents a major step in the project, like design, various manufacturing steps, assembly, etc. There is a natural precedence relation between the project activities. Design must precede manufacturing, which, in turn, is a prerequisite to assembly, and testing. As for the resources, the design activity requires engineers making the blueprint of the parts to be manufactured. The manufacturing steps may require CNC work centers, or milling / turning machines, heat treatment, etc. In the following we assume that the projects are broken down to some main steps, and for each step the key resources are known. So far, we can build a network of activities representing the main steps of the project which in the end delivers the final product to the customer.

Since production plans are made for a longer time horizon, e.g., 26 or 52 weeks, it is desirable that activities are also at the right aggregation level, e.g., the design of the project is represented by a few major design activities, which have a time span of several weeks. Resource are also aggregated, like chef-designers, or a group of identical CNC machines is considered as a single cumulative resource. The processing capacity of a cumulative resource equals the sum of the processing capacities of the resources grouped together.

In practice, the intensity of aggregated activities vary over time. The intensity increases gradually to a maximum level permitted by technological constraints, then it is run for a while at maximum, or close to maximum level, For instance, if a project needs 100 hours of CNC machining, only 10% can be done daily, since several operations must be done on the same part. On the other hand, the activities may overlap

in time. For example, design provides blueprints to manufacturing, and as manufacturing of parts progresses, more and more components can be assembled. Therefore, we connect the activities of the project by *feeding precedence constraints*. Such a constraints specifies that, say, 20% of activity A must be completed before activity B may start, and then B cannot progress faster than A., for an illustration see Figure 1.

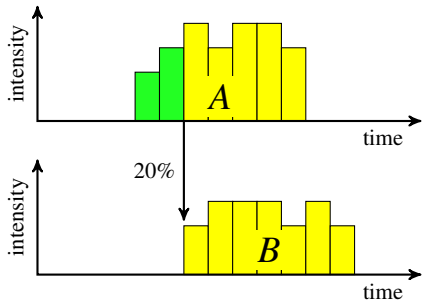

Fig. 1: Variable intensity activities connected by a feeding precedence constraint.

It is natural to assume that resource consumption of variable intensity activities is proportional to their intensity over time. For renewable resources, like machine tools, or workforce, if the intensity of an activity i is x_t^i in time period t, then its resource consumption from some renewable resource R_k is $q_k^i \cdot x_t^i$, where q_k^i is the total demand of activity i from resource k. Of course, one may consider more general functions for computing the resource usage of activities depending on their intensity, however, we consider only linear functions in this paper.

In the rest of this section, we describe a mathematical model and discuss some solution approaches. Finally, we overview some possible extensions of the model.

2.1 Modeling by a mixed-integer linear program

Firstly, we introduce formally the problem data and the objective function, and then describe a mixed-integer linear program for solving it to optimality. There are a set N of activities, and a set RR of renewable resources. The time horizon is divided into time periods $1,\ldots,T$, and any changes in the project can occur only at the border of two consecutive time periods. Each activity i has a time window $[r_i,d_i]$ in which it has to be completed, a maximum intensity $a^i > 0$, and resource requirements $q_k^i \geq 0$ for $k \in RR$. The activities are connected by feeding precedence constraints given by triples (i,j,f_{ij}) meaning that an f_{ij} fraction of activity i must be completed before starting activity j. Each renewable resource $k \in RR$ has a time varying capacity b_{kt}, and an additional external capacity which can be purchased at the expense of additional costs.

The objective is to find an intensity assignment to the activities such that each activity is entirely processed in its time window, the nonrenewable resource constraints are respected, and the cost of external resource usage is minimized. The external resource usage from some renewable resource $k \in RR$ can be measured as the total resource usage above the internal capacities, i.e., $\max\{\sum_{t=1}^{T}\sum_{i\in N} q_k^i x_t^i - b_{kt}, 0\}$. This is illustrated in Figure 2. The motivation for this objective function is the fact that companines usually have their internal workforce and other production capacities from renewable resources, and in case of bottlenecks, they are willing to subcontract or hire some extra resources.

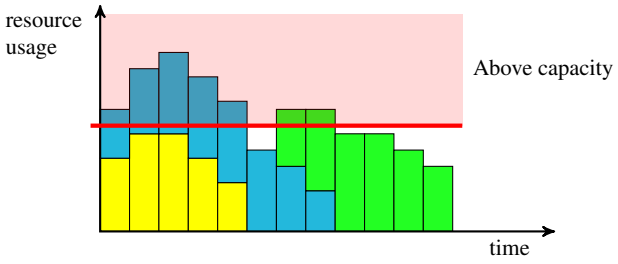

Fig. 2: Resource usage above internal capacity. The internal capacity is indicated by horizontal thick line.

Now we are ready to present a mixed-integer linear program for modelling our pronblem. The decision variables are

$x_t^i =$ intensity of activity i in time period t,

$y_{kt} =$ resoource usage above internal capacity from resource $k \in RR$ in time period t,

$$z_t^{if} = \begin{cases} 0 & \text{if } f\text{-fraction of activity } i \text{ has been completed before time period } t, \\ 1 & \text{otherwise.} \end{cases}$$

The meaning of other symbols in the following problem formulation are as follows:

$\mathcal{N}$ = set of activities,

r^i, d^i = earliest and latest time periods when activity i can be processed,

N_t = set of activities with $r^i \le t \le d^i$,

a^i = maximum intensity of activity i,

c_{kt} = cost of external resource usage from resource $k \in RR$ in time period t,

b_{kt} = internal capacity of resource $k \in RR$ in time period t,

$\bar{b}_{kt}$ = additional external capacity of resource $k \in RR$ in time period t,

E^i = set of precedence relations of the form (i, j, f),

F^i = set of fractions that occur in precedence constraints F^i,

p^{if} = minimum number of periods to finish an f fraction of activity i

$$\min \sum_{t=1}^{T} \sum_{k \in RR} c_{kt} y_{kt} \tag{1}$$

subject to

$$\sum_{t=r^i}^{d^i} x_t^i = 1, \qquad i \in \mathcal{N}, \tag{2}$$

$$\sum_{t=r^i}^{\ell-1} x_t^i \ge f(1 - z_\ell^{if}), \quad \begin{array}{l} i \in \mathcal{N},\ f \in F^i, \\ \ell \in \{r^i + p^{if}, \ldots, d^i\}, \end{array} \tag{3}$$

$$x_t^j \le a^j (1 - z_t^{if}), \qquad i \in \mathcal{N},\ (i, j, f) \in E^i, \tag{4}$$

$$z_t^{if} \ge z_{t+1}^{if}, \qquad \begin{array}{l} i \in \mathcal{N},\ f \in F^i, \\ t \in \{r^i + p^{if}, \ldots, d^i - 1\}, \end{array} \tag{5}$$

$$\sum_{t=r^i}^{\ell} x_t^i \ge \sum_{t=r^j}^{\ell} x_t^j, \qquad \begin{array}{l} (i, j, f) \in E^i, \\ \ell \in \{\max\{r^i, r^j\}, \ldots, \min\{d^i, d^j\}\}, \end{array} \tag{6}$$

$$\sum_{i \in \mathcal{N}_t} q_k^i \cdot x_t^i \le b_{kt} + y_{kt}, \quad k \in RR, t \in \{1, \ldots, T\}, \tag{7}$$

$$0 \le x_t^i \le a^i, \qquad i \in \mathcal{N},\ t \in \{r^i, \ldots, d^i\}, \tag{8}$$

$$0 \le y_{kt} \le \bar{b}_{kt}, \qquad k \in RR,\ t \in \{1, \ldots, T\}, \tag{9}$$

$$z_t^{if} \in \{0, 1\}, \qquad \begin{array}{l} i \in \mathcal{N},\ f \in F^i, \\ t \in \{r^i + p^{if}, \ldots, d^i\}, \end{array} \tag{10}$$

The objective function (1) aims at the minimization of resource hiring/subcontracting costs. Constraints (2) ensure that each activity is totally processed in its time window. The precedence constraints between pairs of activities are described by inequalities (3) through (5). In particular, (3) ensures that z_t^{if} can be zero only if an f-fraction of activity i is completed up to time period $t-1$. Moreover, (4) makes sure that a the successor j of activity i can only start if an f-fraction of activity i is completed. Inequalities (5) ensure that there is only one point in time when the

z_t^{if} switches from 1 to 0. The feeding aspect is captured by (6). The external renewable resource usage is expressed by (7), since y_{kt} is non-negative and it has a positive weight in the objective function, thus strict inequality holds in (7) only if the resource usage is below the internal capacity, in which case $y_{kt} = 0$. Finally, the remaining constraints specify the feasible domains of the variables.

2.2 Solution approaches

There are various exact and heuristic methods for solving the problem (1)-(10). In this section we sketch the main ideas of some of them.

Exact methods: Branch-and-Cut. Branch-and-cut is a kind of branch-and-bound type of method, in which the linear programming relaxation of a MIP is solved and strengthened by inequalities valid for the convex hull of integer solutions, but violated by the (fractional) solution of the LP relaxation [20]. The new inequalities are added to the problem in the root node, and also in search tree nodes. The inequalities are generated by so-called separation procedures, which may be exact or heuristic, and they are designed to find inequalities in a class that are violated by the optimal LP solution of the search tree node.

A special case of the problem in which there can be no overlap between pairs of activities connected by a precedence constraint is discussed in [14]. In that paper, a polyhedral approach is pursued, and an exact branch-and-cut type method is proposed. The crux of the method is a polyhedral characterization of the convex hull of points satisfying (3)-(6), along with (8)-(10). The polyhedral characterization consists of providing the inequalities giving the convex hull of points with *integer* z coordinates satisfying the constraints (3)-(6). The inequalities are used in a branch-and-cut method in which the formulation is strengthened by the new family of inequalities, and they proved very effective in solving the problem with non-overlapping activities to optimality. These results are generalized to overlapping activities in [15]. The computational results for the latter problem show that if we allow more overlap between activities, but we do not change other problem parameters, then the resulting instance is easier to solve. This is plausible, since overlapping of activities connected by a precedence constraint can be seen as a relaxation of the problem without any overlaps between activities connected by precedences.

Exact methods: Branch-and-Price. Branch-and-Price is essentially linear programming based Branch-and-Bound, in which the linear programming relaxation is solved by Column-generation. Column-generation, in turn, is a method for solving linear programs with many, usually millions of variables (columns). In such a method, there is an initial linear program containing only a fraction of the columns of the entire linear program, just enough to have a feasible solution. Then, new columns are inserted gradually using the standard pricing technique of the primal simplex method. The crux of the method is the subroutine for solving the pricing problem efficiently, i.e., given the values of the dual variables associated with the rows of the restricted primal program, one has to find a new column with negative

reduced cost (in case of minimization type of problems), or verify that no such column exists in the full linear program, in which case the current LP basis is optimal. When embedded in a Branch-and-Bound method with appropriate branching rules, we get Branch-and-Price, see Barnhart et al. [3].

Hans [13] proposed a problem formulation amenable for Branch-and-Price. In that formulation, the possible executions of project activities are modeled by a set of binary vectors $\{\beta^h \in \{0,1\}^{|\mathcal{N}|\times T} \mid h \in \Pi\}$, Π being a suitable set of indicies, consisting of the supports of all feasible intensity assignments to the activities. Notice that a binary vector $\beta \in \{0,1\}^{|\mathcal{N}|\times T}$ is the support of a feasible intensity assignment if and only if $\sum_{t=1}^{T} \beta_{i,t} \geq p^{i,1}$, $\min\{t \mid \beta_{i,t} = 1\} \geq r^i$, $\max\{t \mid \beta_{i,t} = 1\} \leq d^i$, and if (i,j) is a pair of activities connected by a precedence relation, then $\max\{t \mid \beta_{i,t} = 1\} < \min\{t \mid \beta_{j,t} = 1\}$. For solving the problem, precisely one vector β^h must be chosen. To this end, Hans introduced new binary variables z_h, $h \in \Pi$, together with the following constraints:

$$\begin{aligned}
&\sum_{h\in\Pi} z_h = 1,\\
&\qquad z_h \in \{0,1\}, \quad h \in \Pi,\\
&0 \leq x_t^i \leq a^i \left(\sum_{h\in\Pi} \beta_{i,t}^h z_h \right), \quad i \in \mathcal{N},\ t \in \{r^i,\ldots,T\}.
\end{aligned}$$

The first two constraints ensure that exactly one vector β^h is chosen. The third one specifies that x_t^i is either 0, or is between 0 and a^i, depending on whether $\beta_{i,t}^h$ is 0 or 1. Hans' model incorporates resource constraints similar to ours, although instead of (9) it has $y_t^k \geq 0$, and $\sum_k y_{kt} \leq \bar{b}_k$, for all t. As the size of Π can be enormous, column generation is the only viable approach to handle this formulation.

Hans proposed various algorithms for solving the pricing problem, and to branch on the right variables. However, the computational results are inferior to those with Branch-and-Cut, see [14] for a comparison.

Heuristics. Heuristical methods usually provide a feasible solution fast, but there is no guarantee for optimality, or even to get solutions close to the optimum. Gademann and Schutten [11] divide the heuristics for our problem into three categories: (i) constructive heuristics, (ii) heuristics that start with infeasible solutions and convert these to feasible ones, and (iii) heuristics that improve feasible solutions.

De Boer and Schutten [7] propose algorithms in the first two categories, and Gademann and Schutten [11] present algorithms that fall in the second and third class. Wullink [21] propose new heuristics and provide a very detailed comparison of the various exact and heuristical methods.

2.3 Extensions and further developments

The model discussed above can be extended in various ways. For instance, in some applications non-renewable resources, like raw materials, or money are to be taken

into account when making feasible project plans. Let NR be the set of non-renewable resources. Each non-renewable resource $k \in NR$ has an initial supply (or stock level) of s_{k0}, and there are further supplies arriving at known moments of time, i.e., for each non-renewable resource $k \in NR$, we have a sequence of u_k supplies with supplied quantities $s_{kh} > 0$ in time points $t_{kh} \in \{1,\dots,T\}$, $h = 1,\dots u_k$. The consumption of activity i from some $k \in NR$ until the end of time period t can be compted as $q_k^i \sum_{\tau=0}^{t} x_\tau^i$.

The following set of constraints can be added to the model (1)-(10):

$$\sum_{i \in N} \sum_{t=1}^{t_{k,(\ell+1)}-1} q_k^i x_t^i \leq s_{k0} + \sum_{h=1}^{\ell} s_{kh}, \quad \ell = 1,\dots,u_k, k \in NR, \tag{11}$$

where we define $t_{k,u_k+1} := T+1$. This constraint ensures that the total amount of resource $k \in NR$ that is used until the $(\ell+1)$-th supply event is not more than the total supply over the first ℓ supply events in addition to the initial stock.

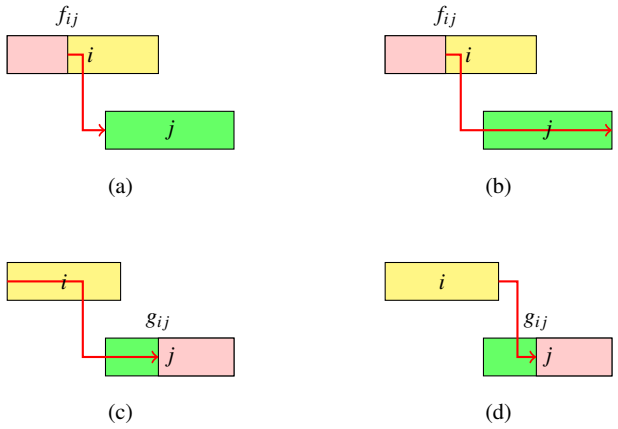

Fig. 3: Illustration of precedence relations: (a) CtS, (b) CtF, (c) StC, (d) FtC.

Another possible direction is to consider further variants of the precedence constraints. In Alfieri et al. [1] and Bianco and Caramia [4] the following 4 types of constraints are considered:

a) %Completed-to-Start (CtS) precedence: successor activity j can start its processing only when, in time period t, the fraction of predecessor activity i that has been processed is greater than or equal to f_{ij} (Fig. 3(a)).
b) %Completed-to-Finish (CtF) precedence: successor activity j can be completed only when, in time period t, the fraction of predecessor activity i that has been processed is greater than or equal to f_{ij} (Fig. 3(b)).

c) Start-to-%Completed (StC) precedence: the fraction execution of successor activity j, in time period t, can be greater than g_{ij} only if the execution of predecessor activity i has already started (Fig. 3(c)).
d) Finish-to-%Completed (FtC) precedence: the fraction execution of successor activity j, in time period t, can be greater than g_{ij} only if the execution of predecessor activity i has been completed (Fig. 3(d)).

The first one in the list is the same as that defined in Section 2.1, while the other three are new. Alfieri et al. provide two problem formulations; in the first one binary variables are used to mark the start and finish of activities over the time horizon, whereas the second one is much like ours, where binary variables are used as execution masks as in Section 2.1. A detailed computational evaluation shows the superiority of the second model in terms of solution time. Bianco and Caramia [4] in turn develop a new Lagrangian relaxation based lower bound for the makespan minimization problem with feeding precedence constraints, where the resource usage is bounded by a constant.

When preemption of activities is not allowed, but a flexible resource usage per activity is desirable, the models discussed above need to be extended by additional constraints to ensure that once an activity started, its intensity does not become zero until it is completely finished. Such formulations are proposed and thoroughly evaluated in Naber and Kolisch [16]. One of their main findings is that the modeling of precedence constraints by the system (3)-(5) is a key ingredient of a strong formulation.

3 Resource leveling

Resource leveling problems aim at finding schedules in which the resource usage is *leveled*, or *smooth* over time. Such problems are well studied in project scheduling (see e.g., [8], [17], [2], [19]), but there are only a few results for machine scheduling problems, see e.g., Rager et al [18]. Notice that in machine scheduling, machines are unary resources that can process one job at a time, while in the more general project scheduling problems each resource can process multiple activities at the same time. Moreover, in project scheduling activities are usually connected by precedence constraints, while in machine scheduling problems this is not always the case [5, 6]. In this section we will study resource leveling problems in a machine environment, where each job is dedicated to a single machine, and may require one or more additional resources whose usage must be leveled.

Consider a scenario where the tasks are already assigned to machines, and the time windows where individual tasks can be processed are already known (e.g. based on precedence constraints, due dates, etc.). Each task may require a given amount of some resources (such as skilled workers, some tools, etc.). For each resource, the available amount is known. Most companies are willing to rent temporary resources (e.g. hiring temporary workers) in order to complete their orders on time, however they want to minimize the extra cost (recall that this was

also the motivation for the objective function in Section 2). Another related application is the classical resource leveling problem: the company wants to minimize the variation of resource usage over time.

Beside the above applications, resource leveling problems occur in scheduling problems where a balanced use of energy is one the main objectives [18], in construction engineering [10], and in production planning [2], [12].

In this section, based on [9], an efficient solution approach is presented for a resource leveling problem in a machine environment as described above. There are m parallel machines, and n_i tasks are assigned to machine i. Preemption of tasks is not allowed, and no machine can process more than one task at a time. Task j has a time window $[e_j, d_j]$ in which it has to be processed for p_j time units. Furthermore there are L types of renewable resources, each resource ℓ having a target level C_ℓ. Task j requires an amount of $b_{j\ell}$ from resource ℓ. An illustrative example is shown on Figure 4.

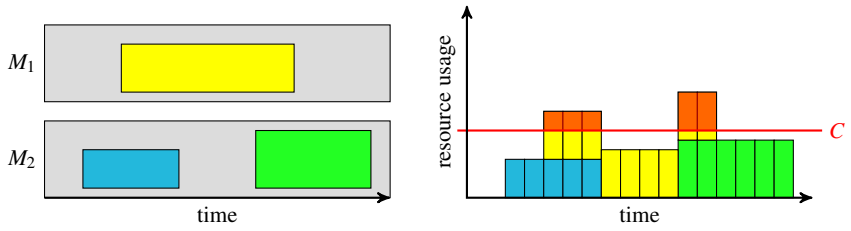

Fig. 4: Example with 2 machines, 3 tasks, and one resource. The height of the tasks is proportional to their resource requirement. The chart on the right hand side shows the resource profile of the schedule for each time unit; the resource overuse is marked above the target level C of the resource.

The goal is to minimize some function of the deviation of the resource usage from pres-pecified values. We consider objective functions of the following form:

$$\sum_{\ell=1}^{L}\sum_{t=0}^{D} f_\ell(y_{\ell t}, C_\ell),$$

where $y_{\ell t}$ is the total usage of resource ℓ at time t, D is the time horizon, and $f_\ell : \mathbb{Q}_+ \times \mathbb{Q}_+ \to \mathbb{Q}_+$ satisfy $f_\ell(x, y-z) = f_\ell(x+z, y)$. Note that f_ℓ may be different for different types of resources.

3.1 Modeling by a mixed integer linear program

The following notations will be used to describe the problem and a solution approach:

$$
\begin{aligned}
m &= \text{number of machines}\\
L &= \text{number of resource types}\\
C_\ell &= \text{target level of resource } \ell\\
J &= \text{set of tasks}\\
J_i &= \text{set of tasks pre-assigned to machine } M_i\\
n_i &= |J_i|, \text{ the number of tasks pre-assigned to machine } i\\
p_j &= \text{processing time of task } j\\
e_j &= \text{release time of task } j\\
d_j &= \text{deadline of task } j\\
b_{j\ell} &= \text{amount of resource } \ell \text{ required by task } j\\
D &= \text{time horizon}
\end{aligned}
$$

The optimization problem can be formulated as a mixed integer linear program:

$$OPT = \min \sum_{\ell=1}^{L} \sum_{t=0}^{D} f_\ell(y_{\ell t}, C_\ell) \tag{12}$$

subject to

$$\sum_{t=0}^{D} x_{jt} = 1, \quad \forall\, j \in J, \tag{13}$$

$$\sum_{j \in J_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} \le 1, \quad \forall\, t \in \{0,\dots,D\},\ i \in \{1,\dots,m\} \tag{14}$$

$$\sum_{j \in J} \sum_{\tau=t-p_j+1}^{t} b_{j\ell} x_{j\tau} - y_{\ell t} = 0, \quad \forall\, t \in \{0,\dots,D\},\ \ell \in \{1,\dots,L\} \tag{15}$$

$$x_{jt} \in \{0,1\}, \quad \forall j \in J,\ t \in \{e_j,\dots,d_j - p_j\}. \tag{16}$$

The decision variables are $x_{jt} \in \{0,1\}$, $j \in J$, $t \in \{e_j,\dots,d_j - p_j\}$, indicating the start times of the tasks, and $y_{\ell t} \in \mathbb{R}$, $j \in J$, $t \in \{0,\dots,D\}$ representing the resource usage in each time period from each resource. As each task must be processed in its time window, if $\tau < e_j$ or $\tau > d_j - p_j$, then $x_{j\tau}$ is not defined and the corresponding term is omitted.

The set of equations (13) ensures that every task is started in precisely one time point $t \in [e_j, d_j - p_j) \cap \mathbb{Z}$. The set of constraints (14) prescribes that no two tasks on the same machine may overlap. Finally, the resource usage is computed in (15).

3.2 Calculation of lower bound by Lagrangian relaxation

In order to compute lower bounds for an optimization problem, a standard technique is to apply Lagrangian duality (see, e.g., [20]), where some nasty constraints of a problem formulation are moved to the objective function so that the resulting problem becomes a *relaxation* of the original, and at the same time is easier to solve.

By dualizing the constraints (15) we obtain the following Lagrangian relaxation of the problem:

$$LB(\boldsymbol{\lambda}) = \sum_{i=1}^{m} LB_i(\boldsymbol{\lambda}) + \min_{y} \sum_{\ell=1}^{L} \sum_{t=0}^{D} \left(f_\ell(y_{\ell t}, C_\ell) - \lambda_{\ell t} y_{\ell t} \right), \tag{17}$$

where

$$LB_i(\boldsymbol{\lambda}) = \min \sum_{\ell=1}^{L} \sum_{t=0}^{D} \sum_{j \in J_i} \sum_{\tau=t-p_j+1}^{t} \lambda_{\ell t} b_{j\ell} x_{j\tau} \tag{18}$$

subject to

$$\sum_{t \in \{e_j, \ldots, d_j - p_j\}} x_{jt} = 1, \quad \forall\, j \in J_i, \tag{19}$$

$$\sum_{j \in J_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} \leq 1, \quad \forall\, t \in \{0, \ldots, D\} \tag{20}$$

$$x_{jt} \in \{0,1\}, \quad \forall j \in J_i,\ t \in \{e_j, \ldots, d_j - p_j\}. \tag{21}$$

From the theory of Lagrangian duality it is known that

$$\max_{\boldsymbol{\lambda}} LB(\boldsymbol{\lambda}) \leq OPT$$

where OPT is the optimum value of (12)-(16).

In [9] it is shown that the subproblems (18)-(21) can be solved efficiently for $f(x,y) := \max\{x-y, 0\}$, and $f(x,y) := (x-y)^2$.

Note that by using this relaxation, the original problem is decomposed into independent single machine problems that can be solved concurrently. Another advantage of this approach is that the structure and the size of the subproblems (identified by (18)-(21)) are independent of the number of resources and the objective function.

3.3 A Branch&Bound method

The nodes in the Branch&Bound search tree represent a constrained version of the original problem, where the time windows of the tasks are narrowed, and the root

node represents the original problem. In each node, the following calculations are performed:

1. **Constraint propagation.** Some well known single machine constraint propagation methods are applied on each machine in order to narrow the time windows of the tasks.
2. **Calculation of lower bound.** By using the subgradient method, the Lagrangian multipliers $\boldsymbol{\lambda}$ are determined for the actual subproblem, and a lower bound is calculated for the actual node. The formulation presented in Section 3.2 is used, however instead of solving the IP-s, their LP-relaxations are considered.
3. **Shaving.** Concurrently for each machine, a shaving procedure is applied. For each task, the lower bound of the objective function is calculated for each possible start time, again using the Lagrangian relaxation. This method may improve the lower bound, and may also narrow the time windows. For an overview of shaving techniques, the reader is referred to [5].
4. **Calculation of upper bound.** By using the solution of the Lagrangian relaxation and applying some heuristics, a solution is sought for the problem represented by the actual node of the Branch&Bound tree.
5. **Branching.** A task is chosen heuristically, and its time window is partitioned into sub-windows. By using the results of shaving, a lower bound can be determined for each child node without extra calculations.

A best-first search is used to traverse the search tree according to the predicted lower bounds of the unvisited nodes, ensuring that the promising combination of time windows are evaluated first. Furthermore, the minimal lower bound among the unvisited nodes represents a lower bound for the original problem.

3.4 Test results

The effectiveness of the presented Branch&Bound method was demonstrated using randomly generated test instances of different sizes, for the following objective functions:

$$f_{\text{lin}}(y_{\ell t}, C_\ell) = w_\ell \max(0, y_{\ell t} - C_\ell) \tag{22}$$

$$f_{\text{quad}}(y_{\ell t}, C_\ell) = (y_{\ell t} - C_\ell)^2 \tag{23}$$

f_{lin} represents the minimization of total weighted resource overuse, while f_{quad} represents the resource leveling problem (i.e. the minimization of the variation of the resource usage over time).

A series of test instances were used with $m = 5, 10, 20$ machines, $t = 10, 15$ and 20 tasks per machine, giving a total of $n = 10m, 15m$ and $20m$ tasks, respectively. Each test class with parameters (m,t) contained 10 instances.

The results of the Branch&Bound method were compared to those obtained by the commercial solver ILOG CPLEX 11.2 using the MIP formulation of the resource

leveling problem (12)-(16). For both programs, a time limit of 1800 seconds was set, and the best lower- and upper bound was recorded at the end of each run.

	m5		**m10**		**m20**		**avg**	
	BB	*CPX*	*BB*	*CPX*	*BB*	*CPX*	*BB*	*CPX*
t10	6.16%	3.10%	0.74%	0.24%	0.36%	0.37%	**2.42%**	**1.24%**
t15	12.94%	11.28%	5.08%	5.96%	0.41%	1.62%	**6.14%**	**6.29%**
t20	18.39%	17.15%	5.41%	7.48%	2.19%	10.39%	**8.66%**	**11.67%**
avg	**12.49%**	**10.51%**	**3.74%**	**4.56%**	**0.99%**	**4.13%**	**5.74%**	**6.40%**

Table 1: Average optimality gap for the linear objective function with $C_\ell = \lfloor \sum_{j \in J} b_{\ell j} p_j / D \rfloor$, and 3 resources.

	m5		**m10**		**m20**		**avg**	
	BB	*CPX*	*BB*	*CPX*	*BB*	*CPX*	*BB*	*CPX*
t10	2.31%	1.51%	0.85%	–	0.24%	–	**1.13%**	–
t15	3.77%	–	1.20%	–	0.60%	–	**1.86%**	–
t20	4.31%	–	2.71%	–	0.37%	–	**2.46%**	–
avg	**3.46%**	–	**1.59%**	–	**0.40%**	–	**1.82%**	–

Table 2: Average optimality gap for the quadratic objective function with $C_\ell = 0$, and 3 resources.

The average optimality gap (defined as $UB/LB - 1$, expressed in percents) in each case is shown in Table 1 and Table 2 for the linear and quadratic objective function, respectively. For the test instances with quadratic objective function, CPLEX was only able to compute lower or upper bounds for the smallest instances within the time limit.

3.5 Computation with multiple CPUs

Our Branch&Bound procedure offers several opportunities for parallel computing. We have investigated the processing of search-tree nodes by multiple CPU cores on a shared-memory computer. Our goal with the tests has been to measure the speed-up that could be gained by parallel processing.

As the type of the problem and the actual test environment (server load, tasks with high priority, etc) may influence the results, the parallel execution was evaluated with two different methods. The first is the ratio of the elapsed CPU time and the wall clock time (see Table 3a). Recall that the notation t10, t15, t20 means that there are 10, 15, and 20, respectively, tasks to be scheduled on a *single* machine, so, in the

cell, say, t20-m20, we provide speed-up for instances with 20×20 tasks (20 tasks per machine). An ideally parallel execution would use $n \cdot t$ CPU seconds with n CPU cores during t seconds wall clock time in an ideal environment. However the wall clock time is still passing even when some CPUs are waiting for synchronizing with the others, and therefore the ratio of total CPU time vs wall clock time is usually worse (smaller) than n.

		m10	m20	avg
t10	**CPU5**	2.65	2.58	**2.62**
	CPU10	3.18	3.24	**3.21**
t15	**CPU5**	3.09	3.17	**3.13**
	CPU10	4.27	4.37	**4.32**
t20	**CPU5**	3.29	3.33	**3.31**
	CPU10	4.81	4.92	**4.87**
avg	**CPU5**	**3.01**	**3.03**	**3.02**
	CPU10	**4.09**	**4.18**	**4.13**

(a) Average ratio of CPU time and wall clock time with different number of CPU cores

		m10	m20	avg
t10	**CPU5**	1.88	2.01	**1.95**
	CPU10	2.14	2.42	**2.28**
t15	**CPU5**	2.77	2.82	**2.79**
	CPU10	3.81	3.79	**3.8**
t20	**CPU5**	3.08	2.91	**3**
	CPU10	4.41	4.2	**4.31**
avg	**CPU5**	**2.58**	**2.58**	**2.58**
	CPU10	**3.45**	**3.47**	**3.46**

(b) Average speed of the algorithm relative to execution with a single CPU

Table 3: Effects of using multiple CPU cores.

The other method is to calculate the average number of nodes evaluated in a second, which can be considered the speed of the algorithm. For each instance the speed of the multi-core test runs was calculated relative to the single-CPU one (see Table 3b). This measure may be less accurate than the previous one because the nodes of the branch-and-bound tree may require different amount of calculation. This is the consequence of the fact that the nodes of the search-tree may represent problems with different complexity.

4 Conclusions

In this chapter we have described a planning and a scheduling problem which have recently gained more and more attention in the academic research, but which frequently occur in practice and need proper solution techniques so that they could be routinely solved by future generation manufacturing planning and scheduling softwares.

We have described some techniques to optimally solve those problems, but the methods mentioned could be turned into heuristics by standard techniques, like truncated branch-and-bound, or beam search.

We believe that variants of these problems do occur in several real-world applications, and a deeper understanding and further work is needed in order to solve them properly in industrial practice.

Acknowledgements This work has been supported by the OTKA grant K112881, and by the NFÜ grant ED_13-2-2013-0002. The research of Tamás Kis has been supported by the János Bólyai research grant BO/00412/12/3 of the Hungarian Academy of Sciences.

References

1. Arianna Alfieri, Tullio Tolio, Marcello Urgo: A Project Scheduling Approach To Production Planning with Feeding Precedence Relations, International Journal of Production Research, **49**, 995–1020 (2011).
2. Francisco Ballestín, Christoph Schwindt, Jürgen Zimmermann: Resource leveling in make-to-order production: Modeling and heuristic solution method, International Journal of Operations Research, **4**, 50–62 (2007).
3. Cynthia Barnhart, Ellis E. Johnson, George E. Nemhauser, Martin W.P. Savelsbergh, Pamela H. Vance: Branch-and-Price: Column generation for solving huge integer programs, Operations Research, **46**, 316–329 (1998) .
4. Lucio Bianco, Massimiliano Caramia: Minimizing the completion time of a project under resource constraints and feeding precedence relations: a Lagrangian relaxation based lower bound, 4OR- Quarterly Journal of Operational Research, **9**, 371–389 (2011).
5. Jacek Blazewicz, Klaus H. Ecker, Erwin Pesch, Günter Schmidt, Jan Weglarz: Handbook on scheduling. From theory to applications. Springer, Berlin, (2007).
6. Peter Brucker: Scheduling Algorithms. Springer, 5th ed. (2007).
7. R. de Boer and J.M.J. Schutten: Multi-project rough-cut capacity planning. In Jalal Ashayeri, William G. Sullivan, and M. Munir Ahmad (Eds.) Flexible Automation and Intelligent Manufacturing, pp. 631–644, New York, Wallingford (U.K.), Begell House, Inc. (1999).
8. Erik L. Demeulemeester, Willy S. Herroelen: Project Scheduling: A Research Handbook, International Series in Operations Research & Management Science, **49**, Kluwer Academic Publishers, (2002).
9. Márton Drótos, Tamás Kis: Resource leveling in a machine environment, European Journal of Operational Research, **212**, 12–21 (2011).
10. Said M. Easa: Resource leveling in construction by optimization, Journal of Construction Engineering and Management, **115**, 302–316 (1998).
11. Noud Gademann, Marco Schutten: Linear-programming-based heuristics for project capacity planning. IIe Transactions, **37**, 153–165 (2005) .
12. Christian Gahm, Bastian Dünnwald, Ramin Sahamie: A multi-criteria master production scheduling approach for special purpose machinery, Int. J.Production Economics, **149**, 89–101 (2014).
13. Erwin W. Hans: *Resource loading by branch-and-price techniques*, Ph.D. thesis, Twente University Press (2001).
14. Tamás Kis: A branch-and-cut algorithm for scheduling of projects with variable intensity activities, Mathematical Programming, **103**, 515–539 (2005).
15. Tamás Kis: RCPS with variable intensity activities and feeding precedence constraints, In: Johanna Józefowska, and Jan Weglarz (Eds.), *Perspectives in Modern Project Scheduling*, pp. 105–129, Springer, New York (2006).
16. Anulark Naber, Rainer Kolisch: MIP models for resource-constrained project scheduling with flexible resource profiles, European Journal of Operational Research, **239**, 335-348 (2014).
17. Klaus Neuamm, Jürgen Zimmermann: Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints, European Journal of Operational Research, **127**, 425–443 (2000).
18. Markus Rager, Christian Gahm, Florian Denz: Energy-oriented scheduling based on Evolutionary Algorithms, Computers & Operations Research, **54**, 218–231 (2015).
19. M. Ranjbar: A path-relinking metaheuristic for the resource levelling problem, Journal of the Operational Research Society **64**, 1071–1078 (2013).

20. Laurence A. Wolsey: Integer Programming, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Inc. (1998).
21. Gerhard Wullink: *Resource loading under uncertainty*, PhD thesis, Twentee Univerity Press (2005).